Zineps – Assessment

Estimated time: 2 hours
Tech Stack: .NET (C#), Azure Functions, SQL, REST APIs, Docker (optional), integrations

📦 Context (Company Simulation)

Zineps is a SaaS platform for managing e-commerce shipping with 40+ carrier integrations (e.g. UPS, DPD, PostNL), invoicing, label generation, and tracking. The codebase is event-driven and uses Azure Functions for asynchronous tasks.

🔧 Part 1 – Carrier Integration (Blazor + .NET)

Task: Build a simulated integration for a fictional carrier called SpeedShip in a Blazor project.

Specs:
• Use an in-memory service or stub (no real API required)
• Use the following (fake) endpoint structure:
• POST /auth/token
• POST /shipments
• GET /labels/{trackingNumber}
• API must support:
• Authentication (token or basic auth)
• Creating a shipment
• Fetching a shipping label (PDF or base64)

Requirements:
• Build a SpeedShipIntegrationService that follows an ICarrierIntegration pattern
• Implement a fallback when authentication fails (e.g. retry or error logging)
• ✅ Bonus: Add 1–2 unit tests to demonstrate testability

💸 Part 2 – Invoice Reconciliation Logic

Scenario: Every Monday, Zineps fetches invoices from carriers and compares them with customer charges to detect discrepancies.

Task:
• Create a function (can be an Azure Function or simple class method) that:
• Accepts two lists: CarrierInvoiceLines and CustomerCharges
• Matches them by trackingNumber
• Flags discrepancies in price, weight, or zone
• Returns a structured DiscrepancyReport (JSON)

Example Structures:
CarrierInvoiceLine: {

```
  trackingNumber: "123ABC",
  amount: 5.99,
  weight: 2.0,
  zone: "NL"
}

CustomerCharge: {
  trackingNumber: "123ABC",
  billedAmount: 6.99,
  declaredWeight: 1.5,
  zone: "NL"
}
```

Bonus:

- Define your own DiscrepancyReport format
- Make the logic extensible (e.g. future fields like fuelSurcharge)
- Describe how you'd scale this if processing **1M+ records per week**

🛠️ Part 3 – Architecture / Infra Questions

Please provide short written answers (1–2 short paragraphs each):

1. How would you set up **Docker + Azure DevOps CI/CD** to auto-deploy the backend?
2. What design patterns would you use to handle **multiple carrier APIs** with varying formats?
3. How would you architect the system to support **50M+ shipments per year** across Europe?

🌱 Bonus (Optional)

🎯 Bonus Question:

In 3–5 sentences, describe the kind of **engineering culture** you thrive in and what you expect from the team around you.

🧪 Bonus Task:

Add a "testMode" flag to the SpeedShipIntegrationService that simulates carrier responses without side effects. Think in terms of abstraction and configuration (e.g. via dependency injection or environment flags).

📁 Submission Instructions

- Send a ZIP file or GitHub repo of your project
- Include a README.md with:
    - Setup instructions
    - Notes about your approach
- Place answers to Part 3 in a separate file ARCHITECTURE.md

✅ Evaluation Criteria

- Code clarity & structure
- Testability and scalability
- API design & integration understanding
- Architecture & problem-solving mindset
- Communication and written clarity